IET Information Security

Special Section: Selected Papers from 17th International Conference on Cryptology and Network Security (CANS)

IET Journals
The Institution of Engineering and Technology

Complexity of statistical attacks on QC-LDPC code-based cryptosystems

ISSN 1751-8709
Received on 22nd August 2019
Accepted on 7th November 2019
E-First on 17th December 2019
doi: 10.1049/iet-ifs.2019.0420
www.ietdl.org

Paolo Santini[1] ✉, *Marco Baldi*[1], *Franco Chiaraluce*[1]

[1]*DII, Università Politecnica delle Marche, Via Brecce Bianche 12, Ancona, Italy*

✉ *E-mail: p.santini@pm.univpm.it*

Abstract: Public-key cryptosystems built on quasi-cyclic (QC) low-density parity-check and moderate-density parity-check codes are promising candidates for post-quantum cryptography, since they are characterised by compact keys and high algorithmic efficiency. The main issue with this kind of system is represented by the fact that, since the decoding procedure is probabilistic, it may leak information about the secret key. In this work, the authors study cryptanalysis procedures that aim at recovering the secret key by exploiting this fact. They identify the phenomenon that is at the basis of these procedures and show that the QC structure plays an important role in the success of these attacks. They use a graph analogy to study the complexity of these attacks, and show that their feasibility strongly depends on the QC structure. They also devise an approach to perform full cryptanalysis by combining an information set decoding algorithm with some partial knowledge about the structure of the secret key.

1 Introduction

Shor's quantum algorithm [1] can solve, in polynomial time, problems like the integer factorisation and the discrete logarithm, upon which systems like Rivest–Shamir–Adleman and Diffie–Hellman are built. Essentially, because of the upcoming advent of quantum computers, the cryptosystems we are using nowadays will not be secure in a not-so-distant future. For this reason, the National Institute of Standards and Technology (NIST) has initiated the process for the evaluation and subsequent standardisation of post-quantum cryptosystems [2], with the aim of finding valid alternatives to quantum-vulnerable though widespread systems.

In this scenario, some candidates of significant interest are those based on coding theory, which was introduced by the seminal works of McEliece [3] and Niederreiter [4]. The security of these schemes is based on the hardness of the so-called Syndrome Decoding Problem (SDP), i.e. the problem of decoding a random linear code, which has been proven to be NP-hard [5]. The best SDP solvers are known as Information Set Decoding (ISD) algorithms that, despite many improvements over time (see [6–8]), still have exponential complexity, even when implemented on quantum computers [9].

The main issue with the classic McEliece cryptosystem, which is based on Goppa codes, is represented by the large size of its public keys, which essentially correspond to some representation of an error-correcting code. One way to address this issue consists of replacing Goppa codes with structured codes, i.e. codes admitting a compact representation through geometrical relations involving the elements in the public key. A common choice is that of using quasi-cyclic (QC) random or pseudo-random codes, without any underlying algebraic structure.

In this paper, we focus on the case of QC low-density parity-check (QC-LDPC) and QC moderate-density parity-check (QC-MDPC) codes, i.e. codes represented by a parity-check matrix that contains a small number of set entries. Such codes are on the basis of two candidates [10, 11] that have been recently admitted to the second round of NIST's competition [12]. Codes of this kind admit a very wide random-like design, and the only known structural attacks to them are those based on the use of ISD for recovering rows of the secret key, which can be seen as low-weight codewords in the dual of the public code. In order to counter these attacks, one must guarantee that the weight of such codewords is not below some security threshold. Such a feature can be obtained by means of a proper transformation matrix, whose effect is that of increasing the minimum distance of the dual code, or by slightly increasing the density of the secret parity-check matrix. In the former case, the secret code is commonly called a QC-LDPC code, while in the latter case, we usually speak of a QC-MDPC code. We point out that, in such a scenario, the QC structure has no consequence in the feasibility of the attacks: ISD algorithms can benefit from a polynomial speed-up [13], which however has no substantial impact on the scheme security.

The main difference between QC-LDPC and QC-MDPC codes with respect to algebraic codes is in the fact that they do not admit efficient bounded-distance decoders. Indeed, all known decoding techniques are probabilistic, in the sense that they might fail with some probability, commonly denoted decryption failure rate (DFR). Such a probabilistic nature of decoding has been shown to leak some information about the secret key. The first-ever published attack of this kind, which is due to Guo *et al.* [14], exploits events of decoding failure to recover the secret key; after that, the same attack procedure has been extended, in order to consider different schemes and other kinds of information leakage [15–19]. Essentially, all these attacks can be divided into two common phases. In the first one, the adversary observes some quantity that is typical of the decoding procedure, such as decoding failures, the number of iterations, the power consumption, and so on. Then, by means of a statistical test, the gathered data is used to recover some characteristic of the secret key. In the second phase, the obtained information is used to reconstruct the secret key, or an equivalent version of it, which guarantees decoding of intercepted ciphertexts. Because of this procedure, all attacks of this kind can be generally denoted as *statistical attacks*. In particular, in [19] we have shown that this statistical analysis can be performed on whichever low-density parity-check (LDPC) or moderate-density parity-check (MDPC) code, and does not depend on the QC structure of the code. In other words, this fact means that the information leakage which is exploited by such attacks is something that intrinsically comes from the decoder, and is not due to the structure of the code.

However, as we show in this paper, the code structure might play an important role in the secret key reconstruction phase. We consider the analogy of this problem with some graph theory problems, and show that the solution of the matrix reconstruction problem is facilitated when the code is QC. To the best of our knowledge, this is the first case in which, for LDPC and MDPC codes, a significant effect of the QC structure on security is

observed. We then review currently known attack procedures and describe them in the general paradigm of statistical attacks.

The remainder of the paper is organised as follows. In Section 2, we describe the notation we use in the paper and recall some concepts about graph theory. In Section 3, we recall the cryptosystems we are considering. In Section 4, we introduce a general framework for statistical attacks, and describe it under a graph perspective. We introduce the problem of the matrix reconstruction by relating it to a graph problem, in which a multigraph needs to be partitioned into proper cliques. In Section 5, we show that, when QC codes are considered, this problem can actually become significantly easier with respect to the general case.

2 Notation

We use bold capital and small letters, respectively, to denote matrices and vectors. Given a matrix $\boldsymbol{A}$, we denote its entry in the ith row and jth column as $a_{i,j}$; given a vector $\boldsymbol{a}$, its ith entry is denoted as a_i. The support of a vector $\boldsymbol{a}$, which is denoted as $\phi(\boldsymbol{a})$, corresponds to the set of indexes pointing at non-null entries in $\boldsymbol{a}$. The Hamming weight of a vector, which is denoted as $\mathrm{wt}(\boldsymbol{a})$, corresponds to the number of its non-null entries, i.e. to the cardinality of its support.

We define a circulant matrix as a square matrix such that each row can be obtained from the first one by applying a shift of one position. Obviously, all rows and columns in a circulant matrix $\boldsymbol{A}$ have the same Hamming weight; then, with some abuse of notation, we will refer to this value as the weight of $\boldsymbol{A}$. In this paper, we will exploit the well-known isomorphism between the ring of $p \times p$ binary circulant matrices and the ring of polynomials $\mathbb{F}_2[x]/(x^p + 1)$, i.e.

$$\boldsymbol{A} = \begin{bmatrix} a_0 & a_1 \cdots & a_{p-1} \\ a_{p-1} & a_0 \cdots & a_{p-2} \\ \vdots & \vdots \ddots & \vdots \\ a_1 & a_2 \cdots & a_0 \end{bmatrix} \leftrightarrow A(x) = \sum_{i=0}^{p-1} a_i x^i .$$

Based on this isomorphism, it can be easily shown that all operations involving circulant matrices can be equivalently described by considering the associated polynomials.

We define a graph $\mathcal{G}$ through a set of vertices V and a set of edges $E \in V^2$. We consider only an undirected graph, i.e. graphs in which an edge is defined just by its endpoints (and not by its direction). In other words, the pairs (i, j) and (j, i) are identical, since they correspond to the same edge. If E contains multiple pairs, then we say that $\mathcal{G}$ is a multigraph; otherwise, $\mathcal{G}$ is called a simple graph.

Let i be a vertex in a graph $\mathcal{G}$: we define its neighbourhood, and denote it as $N_{\mathcal{G}}(i)$, as the set of vertices that are connected to i, i.e. vertices j for which there exists at least an edge $(i, j) \in E$. The degree of a vertex i is denoted as $\deg(i)$ and corresponds to the number of edges that are incident to i.

For a graph $\mathcal{G}$, with vertices set V and edges set E, we say that $\mathcal{G}'$, defined by V' and E', is a subgraph of $\mathcal{G}$, and we write $\mathcal{G}' \subseteq \mathcal{G}$, if $V' \subseteq V$ and $E' \subseteq E$. Given a set $V^* \subseteq V$, we define $\mathcal{G}_{V^*}$ as the subgraph induced by V^*, i.e. the graph whose vertices set is V^* and whose edges are those in $\mathcal{G}$ that connect only vertices in V^*. We say that a subgraph containing w vertices is a w-clique if it is complete, i.e. if each pair of its vertices is connected by (at least) one edge.

Given two graphs $\mathcal{G}_a$ and $\mathcal{G}_b$, with vertices V_a, V_b and edges E_a, E_b, we define $\mathcal{G} = \mathcal{G}_a + \mathcal{G}_b$ as the graph with vertices $V = V_a \cup V_b$ and edges E defined as

$$E = \left\{ (i, j) \in V^2 \middle| (i, j) \in E_a \text{ or } (i, j) \in E_b \right\}. \tag{1}$$

Let i be a vertex in a graph $\mathcal{G}$: then, $\mathcal{G}' = \mathcal{G} \backslash i$ is the graph with vertices $V' = V \backslash i$ and whose edges E' are those in $\mathcal{G}$ that do not have i as an endpoint.

3 System description

In this section, we briefly recall the main principles of public-key encryption schemes and key encapsulation mechanisms based on QC codes with a sparse parity-check matrix. In particular, although our focus is on QC-LDPC codes, we here describe a general framework that encompasses both QC-LDPC [11, 20] and QC-MDPC schemes [10, 21]. As we will see, the QC-MDPC case can be considered as a particular case of the QC-LDPC one, obtainable through a suitable choice of the transformation matrices.

3.1 Key generation

In the schemes, we consider in this paper, the secret (private) key can be written as $K_S = \left\{ \tilde{\boldsymbol{H}}, \boldsymbol{Q} \right\}$. In particular, we have

$$\tilde{\boldsymbol{H}} = \left[\tilde{\boldsymbol{H}}_0 \middle| \tilde{\boldsymbol{H}}_1 \middle| \cdots \middle| \tilde{\boldsymbol{H}}_{n_0-1} \right], \tag{2}$$

where each block $\tilde{\boldsymbol{H}}_i$ is a $p \times p$ circulant matrix, with weight equal to some integer $d_v \ll p$. Usually, n_0 is a small integer while, in order to avoid folding attacks of the type in [22], p is chosen as a prime. The masking matrix $\boldsymbol{Q}$ is an $n \times n$ matrix in QC form (i.e. it is formed by $n_0 \times n_0$ circulant blocks of size p), whose row and column weights are constant and equal to $m \ll n$. The weights of the circulant blocks forming $\boldsymbol{Q}$ can be written in an $n_0 \times n_0$ circulant matrix $\boldsymbol{M}$, such that its element in the ith row and the jth column corresponds to the weight of $\boldsymbol{Q}_{i,j}$. A common choice [11] is that of having $\boldsymbol{M}$ in circulant form; in such a case, we denote its first row as $\boldsymbol{m} = [m_0, m_1, \ldots, m_{n_0-1}]$, and clearly we have $\sum_{i=0}^{n_0-1} m_i = m$.

In order to obtain the public key from the private key, we first compute the matrix $\boldsymbol{H}$ as

$$\boldsymbol{H} = \tilde{\boldsymbol{H}}\boldsymbol{Q} = \left[\boldsymbol{H}_0 \middle| \boldsymbol{H}_1 \middle| \cdots \middle| \boldsymbol{H}_{n_0-1} \right], \tag{3}$$

where each block $\boldsymbol{H}_i$ is again circulant and has weight $\leq m d_v \ll p$. Then, $\boldsymbol{H}$ is the parity-check matrix of a QC code $\mathcal{C}$, with length $n = n_0 p$ and dimension $k = (n_0 - 1)p$, for which the generator matrix in systematic form is obtained as

$$\boldsymbol{G} = \begin{bmatrix} \boldsymbol{I}_{(n_0-1)p} & \left| \begin{matrix} \left(\boldsymbol{H}_{n_0-1}^{-1}\boldsymbol{H}_0\right)^T \\ \left(\boldsymbol{H}_{n_0-1}^{-1}\boldsymbol{H}_1\right)^T \\ \vdots \\ \left(\boldsymbol{H}_{n_0-1}^{-1}\boldsymbol{H}_{n_0-2}\right)^T \end{matrix} \right. \end{bmatrix}, \tag{4}$$

where $\boldsymbol{I}_{(n_0-1)p}$ is the identity matrix of size $(n_0 - 1)p$. The matrix $\boldsymbol{G}$ is then used as the public key K_P. We remark that, when a systematic $\boldsymbol{G}$ is adopted as the secret key, a suitable conversion must be used, in order to achieve indistinguishability under chosen-ciphertext attack (CCA2) [23].

3.2 Encryption

Let $\boldsymbol{u}$ be a k-bit information message to be encrypted, and let $\boldsymbol{e}$ be an n-bit intentional error vector with weight t. The ciphertext $\boldsymbol{c}$ is then obtained as

$$\boldsymbol{c} = \boldsymbol{u}\boldsymbol{G} + \boldsymbol{e}. \tag{5}$$

3.3 Decryption

Decryption starts with the computation of the syndrome as

$$\boldsymbol{s} = \boldsymbol{c}\boldsymbol{Q}^T\tilde{\boldsymbol{H}}^T = \boldsymbol{e}\boldsymbol{Q}^T\tilde{\boldsymbol{H}}^T = \boldsymbol{e}'\tilde{\boldsymbol{H}}^T, \tag{6}$$

which corresponds to the syndrome of the *expanded error vector* $\boldsymbol{e}' = \boldsymbol{e}\boldsymbol{Q}^T$, computed through $\tilde{\boldsymbol{H}}^T$. Then, a syndrome decoding algorithm is applied to $\boldsymbol{s}$, in order to recover $\boldsymbol{e}$. A common choice to decode $\boldsymbol{s}$ is the bit flipping (BF) decoder, firstly introduced in

Input: public key K_P, number of queries $N \in \mathbb{N}$
Output: estimates $\mathbf{A}, \mathbf{B} \in \mathbb{N}^{n \times n}$

$\mathbf{A}, \mathbf{B} \leftarrow \mathbf{0}_{n \times n}$ ▷ Initialization as null arrays
for $i \leftarrow 1$ **to** T **do**
  $\mathbf{c}^{(i)} \leftarrow$ ciphertext obtained through the error vector $\mathbf{e}^{(i)}$
  $y^{(i)} \leftarrow \mathcal{D}(K_S, \mathbf{c}^{(i)})$ ▷ Oracle reply
  $\phi \leftarrow$ support of $\mathbf{e}^{(i)}$
  for $j \in \phi$ **do**
    for $l \in \phi \setminus j$ **do**
      $a_{j,l} \leftarrow a_{j,l} + y^{(i)}$
      $b_{j,l} \leftarrow b_{j,l} + 1$
    end for
  end for
end for
return $\{\mathbf{A}, \mathbf{B}\}$

Fig. 1 *Algorithm 1: GSA*

[24], or one of its variants. In the setting used in QC-LDPC code-based systems, decoding can also be performed through a special algorithm named *Q-decoder* [11], which is a modified version of the classical BF decoder and exploits the fact that $\boldsymbol{e}'$ is obtained as the sum of rows from $\boldsymbol{Q}^T$. The Q-decoding procedure will be briefly described in the next section.

QC-MDPC code-based systems introduced in [21] can be seen as a particular case of the QC-LDPC code-based scheme, corresponding to $\boldsymbol{Q} = \boldsymbol{I}_{n_0 p}$. Encryption and decryption work in the same way, and syndrome decoding is performed through BF on $\tilde{\boldsymbol{H}} = \boldsymbol{H}$. We point out that the classical BF decoder can be considered as a particular case of the Q-decoder, corresponding to $\boldsymbol{Q} = \boldsymbol{I}_{n_0 p}$.

3.4 Q-decoder

Let $\boldsymbol{s}$ be the input syndrome, and define

$$\sigma = \boldsymbol{s} * \boldsymbol{H}, \tag{7}$$

where * denotes the integer inner product. With some straightforward computations, it can be shown that the ith entry of $\boldsymbol{\sigma}$ corresponds to the number of unsatisfied parities in which the ith bit participates. Thus, large values in $\boldsymbol{\sigma}$ are associated with positions that are likely to be affected by errors. This criterion is the one which is applied in a BF decoder, in order to estimate the positions of errors. In the first iteration, $\boldsymbol{s}$ corresponds to the syndrome of the received ciphertext, while in all the other iterations it is obtained by subsequent updates on the initial one.

The novelty of the Q-decoder, with respect to the classical BF decoder, is in the fact that it exploits the knowledge of the matrix $\boldsymbol{Q}$ to improve the decoding performance, a detailed description of this decoding procedure can be found in [11]. In the Q-decoder, decisions about error positions are taken on the basis of some *correlation* values that are computed as

$$\boldsymbol{\rho} = \boldsymbol{s} * \tilde{\boldsymbol{H}} * \boldsymbol{Q} = \boldsymbol{\sigma} * \boldsymbol{Q}\,. \tag{8}$$

As explained in [11, Section 2.5], from the performance standpoint, the Q-decoder approximates a BF decoder working on $\boldsymbol{H} = \tilde{\boldsymbol{H}}\boldsymbol{Q}$. However, by exploiting $\tilde{\boldsymbol{H}}$ and $\boldsymbol{Q}$ separately, the Q-decoder achieves lower complexity than BF decoding working on $\boldsymbol{H}$. The aforementioned performance equivalence is motivated by the following relation:

$$\begin{aligned} \boldsymbol{\rho} &= \boldsymbol{s} * \tilde{\boldsymbol{H}} * \boldsymbol{Q} \\ &= \boldsymbol{e}\boldsymbol{Q}^T\tilde{\boldsymbol{H}}^T * \tilde{\boldsymbol{H}} * \boldsymbol{Q} \\ &= \boldsymbol{e}\boldsymbol{H}^T * \tilde{\boldsymbol{H}} * \boldsymbol{Q} \\ &\simeq \boldsymbol{e}\boldsymbol{H}^T * \boldsymbol{H}, \end{aligned} \tag{9}$$

where the approximation $\tilde{\boldsymbol{H}} * \boldsymbol{Q} \simeq \tilde{\boldsymbol{H}}\boldsymbol{Q} = \boldsymbol{H}$ comes from the sparsity of both $\tilde{\boldsymbol{H}}$ and $\boldsymbol{Q}$. Thus, (9) shows how the decision metric considered in the Q-decoder approximates that used in a BF decoder working on $\boldsymbol{H}$.

4 General statistical attacks

In this section, we describe statistical attacks against schemes in which no masking matrix is used, such as the QC-MDPC McEliece [21]. We first recall the approach introduced in [19], in which a generic model for such attacks is introduced. We then provide an interpretation of the type of information that can be recovered by the opponent and, through a graph analogy, we describe a general approach that allows for the recovery of the secret key. As our analysis highlights, the problem of matrix reconstruction can be related to a well-known hard problem in graph theory. In this section, we do not consider any specific code structure, and leave the analysis of QC codes in Section 5.

4.1 GSA attack

We remind here the attack proposed in [19], that we call the General Statistical Attack (GSA). Such a procedure is, in principle, applicable to any LDPC or MDPC code, regardless of its particular structure.

Let K_S and K_P be a pair of private and public keys, respectively. In a statistical attack, an adversary, who has the availability of K_P, first produces T ciphertexts $\boldsymbol{c}^{(i)}$, for $i = 1, 2, \ldots, T$. He then queries a decryption oracle, which is modeled through an algorithm $\mathcal{D}$ provided with the secret key K_S. The oracle applies the decryption algorithm on the received query, and replies with some quantity that is characteristic of the decoding procedure. The type of reply can be heterogeneous and depends on the particular scenario we are modelling. For instance, it might consist of the decryption outcome (i.e. success or failure), the required time for decoding, the power consumption of some particular step, etc. (see [14, 15, 17–19, 25] for some concrete possibilities). The adversary first collects all the oracle replies, and then performs a statistical analysis on them, with the aim of guessing some information about the secret key.

Essentially, the main idea behind statistical attacks is represented by the fact that the decoding procedure is probabilistic and depends on some relation existing between the error vector and the secret key. Thus, when the error vector is known to the opponent, the decoding procedure intrinsically leaks information about the secret key. In particular, the GSA attack exploits the relation between couple of columns in the secret key and the expected value of the observed quantity.

The main attack procedure is shown in Algorithm 1 (see Fig. 1). The error vector used in the ith query is denoted as $\boldsymbol{e}^{(i)}$, and the corresponding oracle reply is $y^{(i)}$.

The output matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are used by the adversary to compute the matrix $\boldsymbol{D}$, whose entry in position (i, j) is $d_{i,j} = a_{i,j}/b_{i,j}$. It is easy to see that, when T is sufficiently large, we have

$$d_{i,j} \simeq E\big[y \big| e_i = 1,\, e_j = 1\big], \tag{10}$$

where $E[\,\cdot\,]$ denotes the expected value. In other words, the values of $d_{i,j}$ are estimates of the average value of the observed quantity, conditioned to the fact that the error vector contains two fixed set positions. What happens is that $d_{i,j}$ tends to have different distributions, on the basis of the number of common ones between the ith and the jth columns. A heuristic justification of this phenomenon is presented in [19]. Thus, by means of hypothesis tests, the adversary can recover the number of intersections between two columns in the secret key.

4.2 Adjacency matrix

In the previous section, we have described how an adversary can learn the number of overlapping ones between pairs of columns in $\boldsymbol{H}$. These values can then be used to build an $n \times n$ matrix $\boldsymbol{\Lambda}$, whose entries are defined as

$$\lambda_{i,j} = \begin{cases} |\phi(\boldsymbol{h}_i) \cap \phi(\boldsymbol{h}_j)| & \text{if } i \neq j \\ 0 & \text{if } i = j \end{cases}, \tag{11}$$

where $\boldsymbol{h}_i$ denotes the ith column in $\boldsymbol{H}$. The matrix $\boldsymbol{\Lambda}$, already introduced in [19], will be called the *adjacency matrix* of $\boldsymbol{H}$; by construction, $\boldsymbol{\Lambda}$ is clearly symmetric.

In particular, the matrix $\boldsymbol{\Lambda}$ can be associated to a multigraph $\mathcal{G}$ with n vertices, labeled from 0 to $n-1$, and whose edges are defined by the entries of $\boldsymbol{\Lambda}$, such that vertices i and j are connected by $\lambda_{i,j}$ distinct edges. We point out that, because of the choice $\lambda_{i,i} = 0$, $\mathcal{G}$ does not contain loops. Let i be a vertex in $\mathcal{G}$: then, its degree is related to $\boldsymbol{\Lambda}$ through the following equation:

$$\deg(i) = \sum_{j=0}^{n-1} \lambda_{i,j}. \tag{12}$$

In particular, the constructed multigraph $\mathcal{G}$ is associated to the matrix $\boldsymbol{H}$, as the following lemma states.

Lemma 1: Let $\boldsymbol{\Lambda} \in \mathbb{N}^{n \times n}$ be the adjacency matrix of a matrix $\boldsymbol{H} \in \mathbb{F}_2^{r \times n}$, with rank r, and denote with $\mathcal{G}$ the associated multigraph. Then, the following properties hold:

(i) if the ith row of $\boldsymbol{H}$ has weight w_i and support ϕ, then $\mathcal{G}$ contains a clique of size w_i, whose vertices set corresponds to ϕ;
(ii) if the ith column of $\boldsymbol{H}$ has weight v_i, then it is possible to find v_i distinct cliques which involve the vertex i.

Proof: We first prove thesis (i); let us suppose that the ith row of $\boldsymbol{H}$ has weight w_i, and denote its support as ϕ. Each pair of indexes $(j,l) \in \phi^2$, with $j \neq l$, identifies a pair of columns in $\boldsymbol{H}$ that overlap in, at least, one position. Then, we have

$$\lambda_{j,l} \geq 1, \quad \forall (j,l) \in \phi^2,\ j \neq l.$$

Then, by the construction of the multigraph, we have

$$(j,l) \in E, \quad \forall (j,l) \in \phi^2,\ i \neq j.$$

The above equation defines a clique of size equal to the cardinality of ϕ, i.e. w_i.

Proving thesis (ii) is now straightforward: since each row in $\boldsymbol{H}$ is associated with a clique in $\mathcal{G}$, we know that, if the ith column of $\boldsymbol{H}$ has weight v_i, this means that i participates in v_i cliques. The matrix $\boldsymbol{H}$ has rank r, thus, it cannot have two equal rows: then, all the cliques involving i must have a different set of vertices. □

In particular, the following proposition arises as a straightforward consequence of Lemma 1.

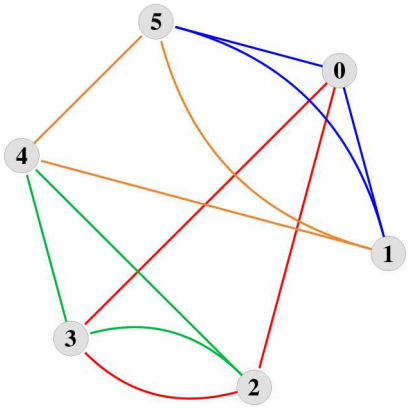

Fig. 2 *Multigraph associated with the example $\Lambda(\boldsymbol{H})$; the cliques partition defined in Proposition 1*

Proposition 1: Let $\boldsymbol{\Lambda} \in \mathbb{N}^{n \times n}$ be the adjacency matrix of a matrix $\boldsymbol{H} \in \mathbb{F}_2^{r \times n}$, with rank r, and denote with $\mathcal{G}$ the associated multigraph. Let w_i and v_j denote, respectively, the weights of the ith row and the jth column of $\boldsymbol{H}$. Then, $\mathcal{G}$ can be partitioned into r cliques $\mathcal{Z}_l$ such that

(i) $\mathcal{Z}_l$ has size w_l, for $l = 0, \ldots, r-1$, and the involved vertices constitute the support of the lth row in $\boldsymbol{H}$;
(ii) if $(i,j) \in E$, then there are $\lambda_{i,j}$ cliques which contain both vertices i and j;
(iii) the vertex j, for $j = 0, 1, \ldots, n-1$, is contained in v_j cliques;
(iv) $\sum_{i=0}^{r-1} \mathcal{Z}_i = \mathcal{G}$.

In particular, the adjacency matrix defines a class of equivalence for matrices over $\mathbb{F}_2$. Indeed, let $\boldsymbol{H}$ be a matrix of size $r \times n$, with adjacency matrix $\boldsymbol{\Lambda}$, and define $\mathrm{Adj}\{\cdot\}$ as the operator that, when applied on a matrix, returns its adjacency matrix. It is easy to see that, for all permutation matrices $\boldsymbol{\Pi} \in \mathbb{F}_2^{r \times r}$, we have $\mathrm{Adj}\{\boldsymbol{\Pi H}\} = \mathrm{Adj}\{\boldsymbol{H}\}$. Then, given an adjacency matrix $\boldsymbol{\Lambda}$ of a full rank matrix $\boldsymbol{H}$, the following result holds:

$$\left|\{\boldsymbol{H}^* \in \mathbb{F}_2^{r \times n} \mid \mathrm{Adj}\{\boldsymbol{H}^*\} = \boldsymbol{\Lambda}\}\right| \geq r!, \tag{13}$$

since $r!$ is equal to the number of distinct $r \times r$ permutations. Obviously, the rows of $\boldsymbol{H}$ must be all different (otherwise, it is $\mathrm{rank}(\boldsymbol{H}) < r$), so distinct permutations will lead to distinct matrices (i.e. they have at least two positions in which the corresponding rows are different).

A very common choice for the cryptosystems we are considering is that of relying on (v,w)-regular codes, i.e. codes that are described by a parity-check matrix that has constant column weight v and constant row weight w. In such a case, all vertices in $\mathcal{G}$ will have the same degree $v(w-1)$; we can then call $\mathcal{G}$ a *regular multigraph*. For the sake of clarity, we show an example of such a multigraph; we consider the following matrix:

$$\boldsymbol{H} = \begin{bmatrix} 1 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 & 0 & 0 \end{bmatrix},$$

which corresponds to the parity-check matrix of a (2, 3)-regular code. Its corresponding multigraph is shown in Fig. 2, where the cliques partition defined in Proposition 1 is emphasised with different colours.

When considering regular codes, the problem of reconstructing a matrix $\boldsymbol{H}$ from its adjacency matrix can be stated in a very natural way. We formulate this problem in the following definition.

Definition 1: (Regular multigraph cliques partition problem): Given $\mathcal{G}$, the multigraph associated to the matrix $\boldsymbol{\Lambda} \in \mathbb{N}^{n \times n}$, which is the adjacency matrix of a parity-check matrix describing a (v,w)-regular code of length n and redundancy r, find a collection of r cliques $\{\mathcal{Z}_0, \ldots, \mathcal{Z}_{r-1}\}$ such that

(i) all cliques have dimension w;
(ii) every node participates in v cliques;
(iii) $\sum_{i=0}^{r-1} \mathcal{Z}_i = \mathcal{G}$.

In other words, the above problem corresponds to finding a partition of $\mathcal{G}$ into r cliques of constant size w. All the cliques must be such that each node is contained in exactly v cliques, and each edge in $\mathcal{G}$ is covered by at least one clique. This problem essentially corresponds to a variant of the *edge clique cover problem*, which is known to be NP-hard [26, 27]. In particular, Gramm *et al.* in [26] propose an algorithm that finds a solution to the edge clique cover problem and whose complexity can be parameterised by the number of searched cliques: if z is the number

Input: adjacency matrix $\mathbf{\Lambda} \in \mathbb{N}^n$
Output: graph $\mathcal{G}$

1: $V, E \leftarrow \varnothing$ ▷ Initialized as empty sets
2: $\phi \leftarrow$ support of the first row of $\mathbf{\Lambda}$
3: $l_\phi \leftarrow |\phi|$
4: $a_0 \leftarrow \phi[0]$ ▷ First element of ϕ
5: **for** $i \leftarrow 2$ **to** l_ϕ **do**
6: **if** $\lambda_{a_0,i} > 0$ **then**
7: $V \leftarrow V \cup i$ ▷ Enlarge the set of vertices
8: **end if**
9: **end for**
10: **for** $i \leftarrow 1$ **to** $l_\phi - 2$ **do**
11: $a_i \leftarrow \phi[i]$ ▷ i-th element of ϕ
12: **for** $j \leftarrow i + 1$ **to** $l_\phi - 1$ **do**
13: $a_j \leftarrow \phi[j]$ ▷ j-th element of ϕ
14: **if** $\lambda_{a_i,a_j} > 0$ **then**
15: $E \leftarrow E \cup (a_i, a_j)$ ▷ Enlarge the set of edges
16: **end if**
17: **end for**
18: **end for**
19: $\mathcal{G} \leftarrow$ graph with vertices set V and edges set E
20: **return** $\{\mathcal{G}\}$

Fig. 3 *Algorithm 2: GraphConstruction*

Input: graph $\mathcal{G}$, clique size u, candidate vertices Z
Output: cliques of size u in $\mathcal{G}$

1: $V \leftarrow$ vertices set of $\mathcal{G}$
2: **for** $i \leftarrow 1$ **to** $|V| + |Z| - u$ **do**
3: $a \leftarrow V[i]$
4: $V^{(a)} \leftarrow N(a)$ ▷ Neighborhood of vertex a
5: $\mathcal{G}_{V^{(a)}} \leftarrow$ subgraph induced by $V^{(a)}$
6: **if** $\left|V^{(a)}\right| == u - |Z| - 1$ **then**
7: **if** $\mathcal{G}_{V^{(a)}}$ is complete **then**
8: **output** $\{Z \cup a \cup V^{(a)}\}$ ▷ $\mathcal{G}_{V^{(a)}}$ is a clique
9: **end if**
10: **else if** $\left|V^{(a)}\right| > u - |Z| - 1$ **then**
11: $Z^{(a)} \leftarrow Z \cup a$
12: RecPruning$(\mathcal{G}_{V^{(a)}}, u, Z^{(a)})$
13: **end if**
14: **end for**

Fig. 4 *Algorithm 3: RecPruning*

of searched cliques, then the algorithm runs in $O(2^z)$ time. We can note that in the cases we are interested in, we have $z = r$. No algorithm with better performances is currently known; furthermore, Cygan *et al.* prove in [27] that the approach due to Gramm *et al.* is essentially optimal.

As mentioned, the problem we are studying is a variant of the edge clique cover problem, since we require that all cliques have the same size and, additionally, that each edge is contained in the same number of cliques. In any case, the well-studied hardness of this problem suggests that even for the variant we are considering, the complexity of an algorithm finding a solution cannot be significantly lower than that of the algorithm proposed by Gramm *et al.*

5 Reconstructing QC codes

In this section, we explain why the problem of the matrix reconstruction, stated in Definition 1, becomes significantly easier in the case of a QC matrix $\boldsymbol{H}$. Indeed, because of the QC structure, finding one of its rows is enough to obtain a full description of the matrix. We show how, in such a case, the matrix reconstruction problem is reduced into that of determining a clique of the proper size in a simple graph. We point out that the work in this section is inspired by that of [14]; however, we here analyse the problem under a graph perspective, and provide an algorithm for the matrix reconstruction, which achieves a lower complexity than that of [14]. We first consider the problem of determining a single row of the parity-check matrix $\boldsymbol{H}$ of regular code, and then revise the GJS attack and its complexity [14]. We justify our claims with both theoretical motivations and results of numerical simulations.

5.1 Finding cliques in simple graphs

The sparsity of a simple graph can be defined with respect to the number of its edges: in general terms, a graph can be defined sparse when such a number is significantly smaller than its allowed maximum value. Many definitions of sparsity are present in the literature, together with examples on how to use such definitions to parameterise the complexity of some graph algorithms; for instance we might mention the concepts of *arboricity* [28], *degeneracy* [29, 30] and *community-degeneracy* [31]. However, in this section we do not consider specific algorithms and describe an approach that, even if designed in a naive fashion, already runs in a complexity that is sufficiently low for the cases we are interested in.

Let $\boldsymbol{H}$ be the parity-check matrix of a (v, w)-regular code, with an adjacency matrix $\mathbf{\Lambda}$. We first consider the following result.

Lemma 2: Let $\boldsymbol{H}$ be the parity-check matrix of a (v, w)-regular code, with an adjacency matrix $\mathbf{\Lambda}$. Let ϕ be the support of the first row of $\mathbf{\Lambda}$, i.e. the set containing all values i for which $\lambda_{0,i} > 0$. Then, the following propositions hold:

(i) there are v rows in $\boldsymbol{H}$ whose support corresponds to a subset of ϕ of cardinality $w - 1$;
(ii) let $a \in \phi$: then, there is (at least) a row in $\boldsymbol{H}$ whose support contains both 0 and a.

Proof: We consider the jth row of $\boldsymbol{H}$, which we denote as $\boldsymbol{h}_j$, and suppose that $h_{j,0} = 1$; since the code is regular, there are exactly v rows with this property. For each element a in the support of $\boldsymbol{h}_j$, we have $h_{j,a} = 1$; since we also have $h_{j,0} = 1$, this means that columns 0 and a overlap in (at least) one position, and thus $\lambda_{0,a} > 0$. In other words, all set positions in $\boldsymbol{h}_j$, apart from 0, are associated to set positions in the first row of $\mathbf{\Lambda}$. Since we have v rows that contain 0 in their support, this means that their support must be subsets of ϕ; this reasoning proves thesis (i).

In particular, for each $a \in \phi$, there must be a row whose support contains both 0 and a, otherwise, it is $\lambda_{0,a} = 0$. Then, because of thesis (i), the support of this row is subsets of ϕ. □

In particular, starting from $\boldsymbol{H}$, we can construct a simple graph, which contains all information about the support of (at least) one row of $\boldsymbol{H}$. To this end, we consider Algorithm 2 (see Fig. 3), which takes as input the adjacency matrix $\mathbf{\Lambda}$ and returns a simple graph $\mathcal{G}$, which contains all vertices i for which $\lambda_{0,i} > 0$ and $\lambda_{0,a_0} > 0$, with $a_0 = \min\{\phi\}$. The edges in $\mathcal{G}$ are defined by elements in $\mathbf{\Lambda}$: two vertices i and j are connected if and only if $\lambda_{i,j} > 0$. Thus, it is easy to see that there must necessarily be a clique in $\mathcal{G}$ having the size $w - 2$ and such that the interested vertices, together with 0 and a_0, constitute the support of a row in $\boldsymbol{H}$.

Then, the problem of recovering a row of $\boldsymbol{H}$ is reduced to that of finding a clique of size $w - 2$ in the simple graph $\mathcal{G}$; as we show in the following, this task can be easily accomplished through a greedy search in the graph. To this end, we can consider Algorithm 3 (see Fig. 4). Essentially, the algorithm tests all possible paths in the graph, by applying a recursive pruning (i.e. extraction of subgraphs), until a complete subgraph with the desired number of vertices is found. At each call, the algorithm starts from a set of candidates Z and a graph $\mathcal{G}$; then, it selects a vertex a (line 3) and computes the subgraph $\mathcal{G}_{V^{(a)}}$ induced by its neighbour vertices. At each new call, the candidates set is enriched with the tested vertex a; the recursive calls go on until a graph with the desired number of vertices is obtained. In the first call to the algorithm, we set $Z = \varnothing$, $u = w - 2$, while $\mathcal{G}$ corresponds to the output of Algorithm 2 (Fig. 3). We remark that, essentially, this algorithm is developed in the same fashion of that given in [14]. However, as we show in the following sections, its time complexity is improved and, more importantly, it allows for finding cliques in whichever case, regardless of the code structure.

Input number of queries $N \in \mathbb{N}$
Output estimates $\mathbf{a}, \mathbf{b} \in \mathbb{N}^{\left\lfloor \frac{p}{2} \right\rfloor}$

$\mathbf{a} \leftarrow$ zero initialized vector of length $\left\lfloor \frac{p}{2} \right\rfloor$
$\mathbf{b} \leftarrow$ zero initialized vector of length $\left\lfloor \frac{p}{2} \right\rfloor$
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{c}^{(i)} \leftarrow$ ciphertext encrypted with the error vector $\mathbf{e}^{(i)}$
  $y^{(i)} \leftarrow \mathcal{D}\left(K_S, \mathbf{c}^{(i)}\right)$ ▷ Oracle reply
  $\mathbf{e}^{(i)}_{n_0-1} \leftarrow$ last length-p block of $\mathbf{e}^{(i)}$
  $\boldsymbol{\Delta}(\mathbf{e}^{(i)}_{n_0-1}) \leftarrow$ distance spectrum of $\mathbf{e}^{(i)}_{n_0-1}$
  for $\left\{ d \in \boldsymbol{\Delta}\left(\mathbf{e}^{(i)}_{n_0-1}\right) \right\}$ **do**
    $b_d \leftarrow b_d + 1$
    $a_d \leftarrow a_d + y^{(i)}$
  end for
end for
return $\{\mathbf{a}, \mathbf{b}\}$

Fig. 5 *Algorithm 4: GJS distance spectrum recover*

We point out that some specific algorithms for clique finding in sparse graphs may also be used (for instance see [30, 31]). It is likely that the use of some specific algorithm will result in improved algorithmic performances; however, even the naive approach is enough to obtain sufficiently low complexities.

5.2 Complexity analysis

In this section, we provide a complexity estimate of Algorithm 3 (Fig. 4), by considering the average number of paths that are tested during the search. First of all, we consider the probability that the support of two columns in $\boldsymbol{H}$ share some common elements. This probability, which estimates the density of $\boldsymbol{\Lambda}$ (i.e. the ratio between the number of set entries and n^2), can be obtained by modelling the columns as random vectors of length r and weight v. This way, we obtain such probability as

$$\rho = 1 - \frac{\binom{r-v}{v}}{\binom{r}{v}}. \tag{14}$$

We now need to consider the number of vertices in $\mathcal{G}$ which is provided as the output of Algorithm 2 (Fig. 3). This graph contains all vertices that are connected to both 0 and a_0, with $a_0 = \min\{\phi\}$. In other words, this means that $\mathcal{G}$ contains all vertices corresponding to columns in $\boldsymbol{H}$ that possess common ones with columns $\boldsymbol{h}_0$ and $\boldsymbol{h}_{a_0}$. Thus, we might estimate the number of vertices in $\mathcal{G}$ as

$$n_{\mathcal{G}} = n\rho^2. \tag{15}$$

It can be verified that, for the parameters we are interested in this paper, we always have $\rho < 1/2$: thus, the number of vertices in $\mathcal{G}$ is significantly smaller than n. Additionally, each two vertices in $\mathcal{G}$ will be connected by an edge with probability approximately equal to ρ: then, the graph $\mathcal{G}$ is somehow sparse, and we can thus expect the execution of Algorithm 3 (Fig. 4) to be quite fast.

We can now evaluate the complexity of running Algorithm 3 (Fig. 4). We assume that all vertices in $\mathcal{G}$ behave as random and independent objects and that each pair of vertices is connected by an edge with probability ρ. At each call, the algorithm prunes the graph, by cutting edges and vertices and returning a subgraph. In particular, in the first call the average number of saved vertices can be estimated as $n_{\mathcal{G}}\rho$. In the second call, we can again consider the input graph as a random graph, with edge probability ρ: thus, the number of vertices in the output graph can be obtained as $n_{\mathcal{G}}\rho^2$. In general, in the jth call, we approximately will have $n_{\mathcal{G}}\alpha^j$ edges. Remember that, at each recursive call, the algorithm adds a vertex to the current candidate set Z. Thus, at the recursive jth call, the set Z contains exactly j vertices; the algorithm will stop pruning if the number of vertices in the produced graph is $\leq u - j$. Thus, the average number of recursive calls can be denoted as $j_{\max}$, and can be estimated as

$$j_{\max} = \min_j \left\{ n_{\mathcal{G}}\rho^j \leq u - j \right\}. \tag{16}$$

The number of paths that is tested by the algorithm can then be estimated by considering all possible combinations of such subgraphs. We consider this number as the complexity of Algorithm 3 (Fig. 4), and denote it as $C_{\mathcal{G}}$; we have

$$C_{\mathcal{G}} = O\left(\prod_{j=1}^{j_{\max}} n_{\mathcal{G}}\rho^j \right) = \left(n_{\mathcal{G}}^{j_{\max}} \rho^{\frac{j_{\max}(j_{\max}+1)}{2}} \right). \tag{17}$$

5.3 GJS attack

We are now ready to revise the attack procedure due to Guo *et al.* [14]. This attack, which is specific to the case of QC codes described by parity-check matrices in the form (2), is based on the concept of *distance spectrum*, defined in the following.

Definition 2: Distance spectrum: Let $\boldsymbol{a}$ be a vector of length p and support ϕ. For a pair of elements $(b_0, b_1) \in \phi^2$, with $b_0 \neq b_1$, we define the corresponding cyclic distance as

$$d = \min\{ \pm (b_0 - b_1) \bmod p \}.$$

The distance spectrum is defined as the set $\boldsymbol{\Delta}(\boldsymbol{a})$ containing all such distances. We say that a distance d has multiplicity $\mu_d^{(\boldsymbol{a})}$ if there are $\mu_d^{(\boldsymbol{a})}$ distinct non-ordered couples of elements of ϕ at a distance d.

Since the distance spectrum is invariant to cyclic shifts, all the rows of a circulant matrix share the same distance spectrum; thus, we can define the distance spectrum of a circulant matrix as the distance spectrum of any of its rows (the first one for the sake of convenience).

It is easy to see that the distance spectrum and the adjacency matrix are actually related. Indeed, let $\boldsymbol{A}$ be a $p \times p$ circulant matrix over $\mathbb{F}_2$, and denote with $\boldsymbol{\Delta}(\boldsymbol{A})$ its distance spectrum and with $\boldsymbol{\Lambda}$ its adjacency matrix. With some straightforward computations [19], it can be shown that $\boldsymbol{\Lambda}$ is circulant as well and, if $d \in \boldsymbol{\Delta}(\boldsymbol{A})$ with multiplicity $\mu_d^{(\boldsymbol{A})}$, then

$$\lambda_{0,d} = \lambda_{0,p-d} = \mu_d^{(\boldsymbol{A})}. \tag{18}$$

As we describe in the following, for a matrix like that in (2), the knowledge of a single circulant block in $\boldsymbol{H}$ is enough to reconstruct the whole matrix from the public key.

In particular, when performing a statistical attack, the adversary can directly focus on the distance spectrum, since it can be recovered by exploiting the QC structure; this can be done by statistical tests on the output of Algorithm 4 (Fig. 5). Indeed, the vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ can be used by the opponent to guess the multiplicity of each distance in the spectrum of $\boldsymbol{H}_{n_0-1}$. The ratios a_d/b_d follow different and distinguishable distributions, with mean values depending on the multiplicity of d. This way, the analysis of the values a_d/b_d allows the opponent to recover $\boldsymbol{\Delta}(\boldsymbol{H}_{n_0-1})$. Once the distance spectrum has been obtained, it can be used to reconstruct the first row of the matrix $\boldsymbol{\Lambda} = \mathrm{Adj}\{\boldsymbol{H}_{n_0-1}\}$. Then, applying Algorithms 2 and 3 (Figs. 3 and 4) will return candidates for one of the rows in $\boldsymbol{H}_{n_0-1}$. The found solution can then be considered as the first row of a circulant $\boldsymbol{H}^*_{n_0-1} = \boldsymbol{\Pi}\boldsymbol{H}_{n_0-1}$, with $\boldsymbol{\Pi}$ being an unknown circulant permutation.

Then, according to (4), the public key can be written as $\boldsymbol{G} = [\boldsymbol{I}|\boldsymbol{P}]$, with

$$\boldsymbol{P} = \begin{bmatrix} \boldsymbol{P}_0 \\ \boldsymbol{P}_1 \\ \vdots \\ \boldsymbol{P}_{n_0-2} \end{bmatrix} = \begin{bmatrix} \left(\boldsymbol{H}_{n_0-1}^{-1}\boldsymbol{H}_0\right)^T \\ \left(\boldsymbol{H}_{n_0-1}^{-1}\boldsymbol{H}_1\right)^T \\ \vdots \\ \left(\boldsymbol{H}_{n_0-1}^{-1}\boldsymbol{H}_{n_0-2}\right)^T \end{bmatrix}. \tag{19}$$

The opponent can then compute the products

$$\boldsymbol{H}_{n_0-1}^{*}\boldsymbol{P}_i^T = \boldsymbol{\Pi}\boldsymbol{H}_i = \boldsymbol{H}_i^{*}, \tag{20}$$

in order to obtain a matrix $\boldsymbol{H}^{*} = \left[\boldsymbol{H}_0^{*}, \boldsymbol{H}_1^{*}, \ldots, \boldsymbol{H}_{n_0-1}^{*}\right] = \boldsymbol{\Pi}\boldsymbol{H}$. This matrix can be used to efficiently decode the intercepted ciphertexts, since

$$\boldsymbol{c}\boldsymbol{H}^{*T} = \boldsymbol{e}\boldsymbol{H}^T\boldsymbol{\Pi}^T = \boldsymbol{s}^T\boldsymbol{\Pi}^T = \boldsymbol{s}^{*T}. \tag{21}$$

Applying a decoding algorithm on $\boldsymbol{s}^{*T}$, with parity-check matrix $\boldsymbol{H}^{*}$, will return $\boldsymbol{e}$ as output. The corresponding plaintext can then be easily recovered by considering the first k positions of $\boldsymbol{c} + \boldsymbol{e}$.

While the GJS attack has been specifically proposed for QC-MDPC cryptosystems, i.e. schemes where no masking matrix is used, it can also be used to attack the case of a QC-LDPC scheme relying on a Q-decoding procedure. In fact, as explained in Section 3.4, the Q-decoder working on $\tilde{\boldsymbol{H}}$ and $\boldsymbol{Q}$ approximates a BF decoder working in $\boldsymbol{H}$: in such a case, the masking matrix has no impact, when considering the GJS attack feasibility. In such a case, the recovered distance spectrum is that of $\boldsymbol{H}_{n_0-1}$ (see (3)), i.e.

$$\boldsymbol{H}_{n_0-1} = \sum_{i=0}^{n_0-1} \tilde{\boldsymbol{H}}_i\boldsymbol{Q}_{i,n_0-1}. \tag{22}$$

In order to justify this claim, we show the results of numerical simulations on code with parameters $n_0 = 2$, $p = 4801$, $d_v = 9$, $m = 5$. The corresponding estimates a_d/b_d, obtained through Algorithm 4 (Fig. 5), are shown in Fig. 6. As we can see from the figure, the distances tend to group into distinct bands, depending on the associated multiplicity in the spectrum.

5.4 GJS complexity

In this section, we evaluate the complexity of performing a complete GJS procedure. First of all, we denote as C_{dist} the number of operations that the opponent must perform, for each decryption query, in order to compute the distance spectrum of $\boldsymbol{e}^{(i)}$ and update the estimates $\boldsymbol{a}$ and $\boldsymbol{b}$. The p-bit block $\boldsymbol{e}_{n_0-1}^{(i)}$ can have a weight between 0 and t; let us suppose that its weight is t_p, which occurs with probability

$$p_{t_p} = \frac{\binom{p}{t_p}\binom{n-p}{t-t_p}}{\binom{n}{t}}. \tag{23}$$

We can assume that in $\boldsymbol{e}_{n_0-1}$ there are no distances with multiplicity ≥ 2 (which is reasonable when $\boldsymbol{e}$ is sparse). The average number of distances in $\boldsymbol{e}_{n_0-1}$ can thus be estimated as $\sum_{t_p=0}^{t} p_{t_p}\binom{t_p}{2}$, which also gives the number of operations needed to obtain the spectrum of $\boldsymbol{e}_{n_0-1}$. Each of these distances is associated with two additional operations: the update of $\boldsymbol{b}$, which is performed for each decryption query, and the update of $\boldsymbol{a}$, which is performed only in the case of a decryption failure. Thus, if we denote as ϵ the DFR of the system and as C_{enc} the complexities of the encryption procedure, respectively, the average complexity of each decryption query can be estimated as

$$C_q = C_{\text{enc}} + (2+\epsilon)\sum_{t_p=0}^{t} p_{t_p}\binom{t_p}{2}. \tag{24}$$

Thus, the complexity of running Algorithm 4 (Fig. 5) can be obtained as

$$C_N \simeq N \cdot C_q = N \cdot \left[C_{\text{enc}} + (2+\epsilon)\sum_{t_p=0}^{t} p_{t_p}\binom{t_p}{2}\right]. \tag{25}$$

Once the distance spectrum has been obtained, the adversary can proceed in the reconstruction of the block $\boldsymbol{H}_{n_0-1}$. The complexity of such a procedure can be estimated by means of (17), by considering $\boldsymbol{H}_{n_0-1}$ as the parity-check matrix of a regular pseudo-code (i.e. with redundancy equal to the code length), with $v = w = md_v$. The total complexity of the attack, measured in terms of a work factor, can then be obtained as

$$WF_{\text{GJS}} = C_N + C_{\mathcal{G}}. \tag{26}$$

In particular, for parameters of practical interest [11, 21], the values of C_N are always significantly below the security level. We now consider the values of C_N: the minimum values of N which allow for the full spectrum reconstruction are complex to estimate. In any case, all numerical simulations reported in the literature have shown how such a goal can be achieved with values of N that are significantly below the security level. In the end, when considering statistical attacks based on decryption failure, it seems safe to assume that, when the DFR is non-negligible, the corresponding values of N are $\ll 2^{SL}$, where SL denotes the security level of the scheme, expressed in bits. When dealing with other kinds of statistical attacks (like timing attacks), it would be surprising if the complexity values C_N are above the security level. Again, the results of all numerical simulations are coherent with this claim.

6 Effect of the masking matrix

In this section, we take into account attacks tailored at schemes in which $\boldsymbol{Q} \neq \boldsymbol{I}_{n_0 p}$ is used and the decoding procedure tries to correct the expanded error vector $\boldsymbol{e}' = \boldsymbol{e}\boldsymbol{Q}$ through the parity-check matrix $\tilde{\boldsymbol{H}}$. With this choice, the GJS attack cannot be performed anymore. In order to justify this fact, let us consider the generic expression of a block of $\boldsymbol{H}$, say the first one

$$\boldsymbol{H}_0 = \sum_{j=0}^{n_0-1} \boldsymbol{Q}_{j,0}\tilde{\boldsymbol{H}}_j = \sum_{j=0}^{n_0-1} \boldsymbol{A}_j. \tag{27}$$

Each $\boldsymbol{A}_j$ can be seen as a sum of cyclically shifted versions of $\tilde{\boldsymbol{H}}_j$ (resp. $\boldsymbol{Q}_{j,0}$) placed at positions depending on the support of the first row of $\boldsymbol{Q}_{j,0}$ (resp. $\tilde{\boldsymbol{H}}_j$). Since all these matrices are sparse, the expected number of cancellations occurring in such a sum is small. This means that, with high probability, distances in $\boldsymbol{Q}_{j,0}$ or $\tilde{\boldsymbol{H}}_j$ are

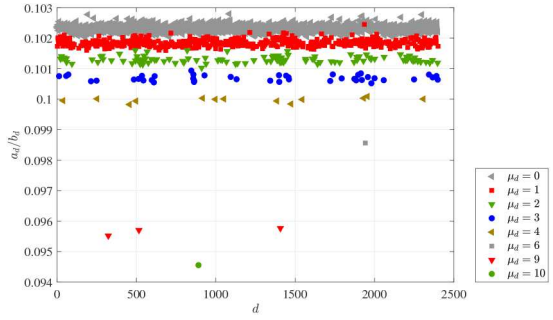

Fig. 6 *Distribution of the opponent's estimates for a QC-LDPC code-based system instance with $n_0 = 2$, $p = 4801$, $d_v = 9$, $[m_0, m_1] = [2, 3]$, $t = 95$, decoded through the Q-decoder. The estimates a_d/b_d correspond to the output of Algorithm 4 (Fig. 5)*

Input number of queries $N \in \mathbb{N}$
Output estimates $\mathbf{a}, \mathbf{b}, \mathbf{u}, \mathbf{v} \in \mathbb{N}^{\lfloor \frac{p}{2} \rfloor}$

$\mathbf{a} \leftarrow$ zero initialized vector of length $\lfloor \frac{p}{2} \rfloor$
$\mathbf{b} \leftarrow$ zero initialized vector of length $\lfloor \frac{p}{2} \rfloor$
$\mathbf{u} \leftarrow$ zero initialized vector of length $\lfloor \frac{p}{2} \rfloor$
$\mathbf{v} \leftarrow$ zero initialized vector of length $\lfloor \frac{p}{2} \rfloor$
for $i \leftarrow 1$ **to** N **do**
  $\mathbf{c}^{(i)} \leftarrow$ ciphertext encrypted with the error vector $\mathbf{e}^{(i)}$
  for $j \leftarrow 0$ **to** $n_0 - 1$ **do**
    $\boldsymbol{\Delta}\left(\mathbf{e}_j^{(i)}\right) \leftarrow$ distance spectrum of $\mathbf{e}_j^{(i)}$
  end for
  $\boldsymbol{\Delta}\left(\mathbf{e}^{(i)}\right) = \bigcup_{j=0}^{n_0-1} \boldsymbol{\Delta}\left(\mathbf{e}_j^{(i)}\right)$
  for $\left\{d \in \boldsymbol{\Delta}\left(\mathbf{e}^{(i)}\right)\right\}$ **do**
    $b_d \leftarrow b_d + 1$
    $a_d \leftarrow a_d + y^{(i)}$
  end for
  for $d \in \boldsymbol{\Delta}\left(\mathbf{e}_{n_0-1}^{(i)}\right)$ **do**
    $v_d \leftarrow v_d + 1$
    $u_d \leftarrow u_d + y^{(i)}$
  end for
end for

Fig. 7 *Algorithm 5: FHS + distance spectrum recovery*

Input number of queries $N \in \mathbb{N}$
Output estimates $\mathbf{a}^{(0)}, \cdots, \mathbf{a}^{(n_0-1)} \in \mathbb{N}^{\lfloor \frac{p}{2} \rfloor}$, $\mathbf{b}^{(0)}, \cdots, \mathbf{b}^{(n_0-1)} \in \mathbb{N}^{\lfloor \frac{p}{2} \rfloor}$

for $j \leftarrow 0$ **to** $n_0 - 1$ **do**
  $\mathbf{a}^{(j)} \leftarrow$ zero initialized vector of length $\lfloor \frac{p}{2} \rfloor$
  $\mathbf{b}^{(j)} \leftarrow$ zero initialized vector of length $\lfloor \frac{p}{2} \rfloor$
end for
for $i \leftarrow 0$ **to** M **do**
  $\mathbf{c}^{(i)} \leftarrow$ ciphertext encrypted with the error vector $\mathbf{e}^{(i)}$
  for $j \leftarrow 0$ **to** $n_0 - 1$ **do**
    $\boldsymbol{\Delta}\left(\mathbf{e}_j^{(i)}\right) \leftarrow$ distance spectrum of $\mathbf{e}_j^{(i)}$
    for $d \in \boldsymbol{\Delta}\left(\mathbf{e}_j^{(i)}\right)$ **do**
      $b_d^{(j)} \leftarrow b_d + 1$
      $a_d^{(j)} \leftarrow a_d + y^{(i)}$
    end for
  end for
end for

Fig. 8 *Algorithm 6: FHZ distance spectrum recovery*

present also in $\boldsymbol{A}_j$. Since the BF decoder performance depends on distances in both $\tilde{\boldsymbol{H}}$ and $\boldsymbol{Q}$, the opponent can correctly identify these distances by analysing Bob's reactions. However, the spectrum of $\boldsymbol{H}_0$ also contains a new set of *inter-block* distances, i.e. distances formed by one entry of $\boldsymbol{A}_i$ and one entry of $\boldsymbol{A}_j$, with $i \neq j$. These distances cannot be revealed by the opponent, because they do not affect the decoding performance when a BF decoder working on the private code is used.

For this reason, in such a case, we must consider attacks that are specific to this situation. In the remaining portion of this section, we revise the attack procedures described in [15, 16].

6.1 FHS+ attack

More recently, a reaction attack specifically tailored to QC-LDPC code-based systems has been proposed in [15]: this attack, differently from the GJS one, takes into account the effect of the matrix $\boldsymbol{Q}$ on the decoding procedure. We refer to this attack as the FHS+ attack. The collection phase in the FHS$^+$ attack is performed through Algorithm 5 (see Fig. 7). We point out that the algorithm we propose is a slightly different (and improved) version of the attack in [15].

As in the GJS attack, the estimates a_d/b_d are then used by the opponent to guess the distances appearing in the blocks of $\tilde{\boldsymbol{H}}$. In particular, every block $\boldsymbol{e}_j^{(i)}$ gets multiplied by all the blocks $\tilde{\boldsymbol{H}}_j$, so the analysis based on a_d/b_d reveals $\boldsymbol{\Delta}(\tilde{\boldsymbol{H}}) = \bigcup_{j=0}^{n_0-1} \boldsymbol{\Delta}(\tilde{\boldsymbol{H}}_j)$. In the same way, the estimates u_d/v_d are used to guess the distances appearing in the blocks belonging to the last block row of $\boldsymbol{Q}^T$. Indeed, the block $\boldsymbol{e}_{n_0-1}^{(i)}$ gets multiplied by all the blocks $\boldsymbol{Q}_{j,n_0-1}^T$. Since a circulant matrix and its transpose share the same distance spectrum, the opponent is indeed guessing distances in the first block column of $\boldsymbol{Q}$. In other words, the analysis based on u_d/v_d reveals $\boldsymbol{\Delta}(\boldsymbol{Q}) = \bigcup_{j=0}^{n_0-1} \boldsymbol{\Delta}(\boldsymbol{Q}_{j,n_0-1})$.

In such a case, the adversary is then provided with information that can be considered as the superimposition of the adjacency matrices of different circulant blocks. In other words, let us first consider $\boldsymbol{\Delta}(\tilde{\boldsymbol{H}})$; by means of (18), the opponent can construct a $p \times p$ adjacency matrix $\boldsymbol{\Lambda}$, which is in QC form and whose first row is such that

$$\lambda_{0,d} = \lambda_{0,p-d} \quad \text{iff} \quad d \in \boldsymbol{\Delta}(\tilde{\boldsymbol{H}}). \tag{28}$$

Then, this matrix can be used as input of Algorithm 2 (Fig. 3), in order to obtain the graph $\mathcal{G}$ in which cliques of size $d_v - 2$ represent rows of the circulant blocks in $\tilde{\boldsymbol{H}}$. The same procedure can be applied on $\boldsymbol{\Delta}(\boldsymbol{Q})$, in order to recover rows of the circulant blocks in the first column layer of $\boldsymbol{Q}$. Again, it is easy to see that, for all the corresponding clique sizes, Algorithm 3 (Fig. 4) runs with very low complexity. Then, in the attack complexity, we can neglect the complexity of the cliques search.

The complexity of this attack, which has been extensively studied in [32], can then be estimated as

$$WF_{\text{FHS}^+} \geq 2^{2n_0 - n^{(1)} - n^{(2)}} \frac{n_0!}{n^{(1)}!} p^{n_0} \log_2(p), \tag{29}$$

where $n^{(1)}$ and $n^{(2)}$ correspond to the number of values in the first row of $\boldsymbol{M}$ that are equal to 1 and 2, respectively.

6.2 FHZ attack

The FHZ attack has been proposed in [16], and is another attack procedure specifically tailored to QC-LDPC code-based systems. The attack starts from the assumption that the number of decryption queries to the oracle is properly bounded, such that the opponent cannot recover the spectrum of $\tilde{\boldsymbol{H}}$ (i.e. the design criterion followed by the authors of LEDApkc [33]). However, it may happen that such a bounded amount of ciphertexts is enough for recovering the spectrum of $\boldsymbol{Q}$: in such a case, the opponent might succeed in reconstructing a shifted version of $\boldsymbol{H}$, with the help of ISD. The distance spectrum recovery procedure for this attack is described in Algorithm 6 (Fig. 8).

The estimates $a_d^{(i)}/b_d^{(i)}$ are then used to guess distances in $\bigcup_{j=0}^{n_0-1} \boldsymbol{\Delta}(\boldsymbol{Q}_{j,i})$; again, as for the FHS+ attack, finding the corresponding cliques in the graph can be done with very low complexity.

The found cliques are then used to build sets of candidates for $\boldsymbol{Q}^T$; in particular, the number of candidates equals

$$N_Q = 2^{n_0^2 - n_0 n^{(2)} - n_0 n^{(1)}} \left\lfloor \prod_{\substack{i=0 \\ \hat{m}_i \geq 2}}^{z} j_i! \right\rfloor^{n_0}, \tag{30}$$

with $n^{(2)}$ and $n^{(1)}$ being the number of entries of the first row of $\mathbf{M}$ that are equal to 2 and 1, respectively, and $[\hat{m}_0, \hat{m}_1, \cdots,$ being the distinct values in the first row of $\mathbf{M}$. The adversary can then use the so obtained candidates to compute matrices $\boldsymbol{G}' = \boldsymbol{G}\boldsymbol{Q}^T$, where $\boldsymbol{G}$ corresponds to the public key. It can be easily shown that $\boldsymbol{G}'$ is a generator matrix of the secret code. Thus, an opponent can apply an ISD algorithm to search for vectors with weight $n_0 d_v$, denoted as

$\bar{v}(x)$ in polynomial notation, such that $\boldsymbol{G}(x)\bar{v}^T(x) = 0$. In particular, some considerations on $\boldsymbol{G}'$ can be made in order to ease the application of ISD; details on this procedure can be found in [32]. As for the FHS+ attack, unless the DFR of the system is significantly low, we can neglect the complexity of Algorithm 6 (Fig. 8), and estimate the complexity of the FHZ attack as

$$WF_{\mathrm{FHZ}} = N_Q \cdot N_G \cdot C_{\mathrm{ISD}}(n_0 p, p, n_0 d_v), \tag{31}$$

where

$$N_G = \left(p^{n_0}\right)^{n_0 - 1} = p^{n_0^2 - n_0}. \tag{32}$$

7 Conclusions

We have reviewed currently known statistical attacks against QC-LDPC and QC-MDPC code-based cryptosystems and introduced a general framework able to encompass them all. In their general form, these attacks are characterised by two phases. The first phase is the collection phase, and is relatively simple to model for a given cryptosystem. The second phase is the matrix reconstruction phase, and is more difficult to model analytically. We have exploited a graph theory approach to model such a phase. This has allowed us to derive closed-form formulas for the overall complexity of these attacks, which are useful for the design of QC-LDPC and QC-MDPC code-based cryptosystems.

8 References

[1] Shor, P.W.: 'Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer', *SIAM J. Comput.*, 1997, **26**, (5), pp. 1484–1509

[2] National Institute of Standards and Technology: 'Post-quantum crypto project', Available at: http://csrc.nist.gov/groups/ST/post-quantum-crypto/, 2016

[3] McEliece, R.J.: 'A public-key cryptosystem based on algebraic coding theory', DSN Progress Report, 1978, pp. 114–116

[4] Niederreiter, H.: 'Knapsack-type cryptosystems and algebraic coding theory', *Probl. Control Inf. Theory*, 1986, **15**, pp. 159–166

[5] Berlekamp, E., McEliece, R., van Tilborg, H.: 'On the inherent intractability of certain coding problems', *IEEE Trans. Inf. Theory*, 1978, **24**, (3), pp. 384–386

[6] Prange, E.: 'The use of information sets in decoding cyclic codes', *IRE Trans. Inf. Theory*, 1962, **8**, (5), pp. 5–9

[7] Stern, J.: 'A method for finding codewords of small weight'. Coding Theory and Applications, Toulon, France, 1989 (LNCS, **388**), pp. 106–113

[8] Becker, A., Joux, A., May, A., *et al.*: 'Decoding random binary linear codes in $2^{(n/20)}$: how $1 + 1 = 0$ improves information set decoding'. Advances in Cryptology - EUROCRYPT 2012, Cambridge, UK, 2012 (LNCS, **7237**), pp. 520–536

[9] Bernstein, D.J.: 'Grover vs. McEliece'. PQCrypto, Darmstadt, Germany, 2010 (LNCS, **6061**), pp. 73–80

[10] Aragon, N., Barreto, P.S.L.M., Bettaieb, S., *et al.*: 'BIKE: bit flipping key encapsulation', NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms, Available at: http://bikesuite.org/, 2017

[11] Baldi, M., Barenghi, A., Chiaraluce, F., *et al.*: 'LEDAkem: a post-quantum key encapsulation mechanism based on QC-LDPC codes'. Post-Quantum Cryptography, Cham, 2018 (LNCS, **10786**), pp. 3–24

[12] Alagic, G., Alperin-Sheriff, J., Apon, D., *et al.*: 'Status report on the first round of the NIST post-quantum cryptography standardization process', National Institute of Standards and Technology, 2019, NISTIR 8240

[13] Sendrier, N.: 'Decoding one out of many'. Post-Quantum Cryptography, Taipei, Taiwan, 2011 (LNCS, **7071**), pp. 51–67

[14] Guo, Q., Johansson, T., Stankovski, P.: 'A key recovery attack on MDPC with CCA security using decoding errors'. ASIACRYPT 2016, Hanoi, Vietnam, 2016 (LNCS, **10031**), pp. 789–815

[15] Fabšč, T., Hromada, V., Stankovski, P., *et al.*: 'A reaction attack on the QC-LDPC McEliece cryptosystem'. Post-Quantum Cryptography: 8th Int. Workshop, PQCrypto 2017, Utrecht, the Netherlands, 2017, pp. 51–68

[16] Fabšič, T., Hromada, V., Zajac, P.: 'A reaction attack on ledapkc'. (2018, https://eprint.iacr.org/2018/140. Cryptology ePrint Archive, Report 2018/140

[17] Eaton, E., Lequesne, M., Parent, A., *et al.*: 'QC-MDPC: A timing attack and a CCA2 KEM'. Post-Quantum Cryptography – 9th Int. Conf., PQCrypto 2018, Fort Lauderdale, FL, USA, 9–11 April 2018, pp. 47–76

[18] Nilsson, A., Johansson, T., Stankovski, P.: 'Error amplification in code-based cryptography', *IACR Trans. Cryptographic Hardware Embedded Syst.*, 2018, **2019**, (1), pp. 238–258

[19] Santini, P., Battaglioni, M., Chiaraluce, F., *et al.*: 'Analysis of Reaction and Timing Attacks Against Cryptosystems Based on Sparse Parity-Check Codes', in Baldi, M., Persichetti, E., Santini, P., *et al.* (Eds.): '*Code-Based Cryptography. CBC 2019. Lecture Notes in Computer Science*' vol **11666** (Springer, Cham, 2019)

[20] Baldi, M., Bodrato, M., Chiaraluce, F.: 'A new analysis of the McEliece cryptosystem based on QC-LDPC codes'. Security and Cryptography for Networks, Amalfi, Italy, 2008 (LNCS, **5229**), pp. 246–262

[21] Misoczki, R., Tillich, J.P., Sendrier, N., *et al.*: 'MDPC-McEliece: new McEliece variants from moderate density parity-check codes'. 2013 IEEE Int. Symp. on Information Theory, Istambul, Turkey, 2013, pp. 2069–2073

[22] Shooshtari, M.K., Ahmadian Attari, M., Johansson, T., *et al.*: 'Cryptanalysis of McEliece cryptosystem variants based on quasi-cyclic low-density parity check codes', *IET Inf. Sec.*, 2016, **10**, (4), pp. 194–202

[23] Kobara, K., Imai, H.: 'Semantically secure McEliece public-key cryptosystems - conversions for McEliece PKC'. Public Key Cryptography, PKC 2001, Cheju Island, Republic of Korea, 2001 (LNCS, **1992**), pp. 19–35

[24] Gallager, R.G.: '*Low-density parity-check codes*' (MIT Press, 1963)

[25] Fabšič, T., Gallo, O., Hromada, V.: 'Simple power analysis attack on the QC-LDPC McEliece cryptosystem', *Tatra Mountains Math. Pub.*, 2016, **67**, (1), pp. 85–92

[26] Gramm, J., Guo, J., Hüffner, F., *et al.*: 'Data reduction and exact algorithms for clique cover', *J. Exp. Algorith.*, 2009, **13**, pp. 2:2.2–2:2.15. Available at: http://doi.acm.org/10.1145/1412228.1412236

[27] Cygan, M., Pilipczuk, M., Pilipczuk, M.: 'Known algorithms for edge clique cover are probably optimal', *SIAM J. Comput.*, 2016, **45**, pp. 67–83

[28] Chiba, N., Nishizeki, T.: 'Arboricity and subgraph listing algorithms', *SIAM J. Comput.*, 1985, **14**, pp. 210–223

[29] Lick, D.R., White, A.T.: 'k-degenerate graphs', *Can. J. Math.*, 1970, **22**, (5), pp. 1082–1096

[30] Eppstein, D., Löffler, M., Strash, D.: 'Listing all maximal cliques in sparse graphs in near-optimal time'. Algorithms and Computation, Berlin, Heidelberg, 2010, pp. 403–414

[31] Buchanan, A., Walteros, J.L., Butenko, S., *et al.*: 'Solving maximum clique in sparse graphs: an $O(nm + n2^{d/4})$ algorithm for d-degenerate graphs', *Opt. Lett.*, 2014, **8**, (5), pp. 1611–1617

[32] Santini, P., Baldi, M., Chiaraluce, F.: 'Assessing and countering reaction attacks against post-quantum public-key cryptosystems based on QC-LDPC codes'. Cryptology and Network Security, Cham, 2018, pp. 323–343

[33] Baldi, M., Barenghi, A., Chiaraluce, F., *et al.*: 'LEDApkc: low dEnsity coDebAsed public key cryptosystem', NIST Post-Quantum Cryptography Project: First Round Candidate Algorithms, Available at: https://www.ledacrypt.org/, 2017